SHREE H. N. SHUKLA GROUP OF COLLEGES

CHAPTER-3

Data Control and Transaction Control Command Introduction to PL/SQL

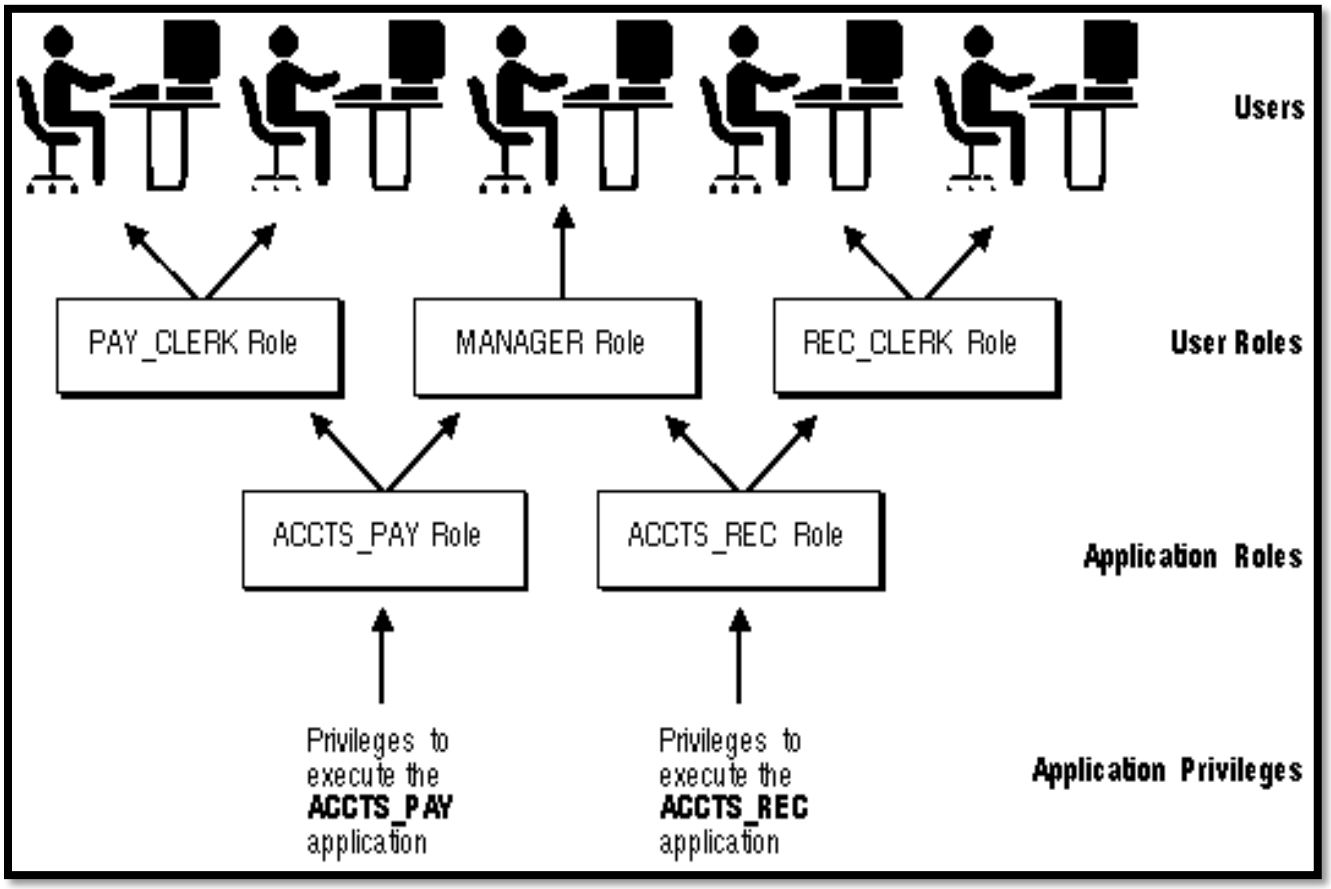
- Grant, Revoke
- Role
- Creating Users
- What is transaction?
- Starting and Ending of Transaction
- Commit, Rollback, Savepoint
- SQL v/s PL/SQL
- PL/SQL Block Structure
- Language construct of PL/SQL (Variables, Basic and Composite Data type, Conditions, looping etc.)
- %TYPE and %ROWTYPE
- Using Cursor(Implicit, Explicit)

SHREE H. N. SHUKLA GROUP OF COLLEGES

Q-1 Write note On Role.

Detail :-

- Oracle provides for easy and controlled privilege management through roles.
- *Roles* are named groups of related privileges that you grant to users or other roles.



- Roles are designed to ease the administration of end-user system and object privileges.

Common Uses for Roles:-

- In general, you create a role to serve one of two purposes:
  - To manage the privileges for a database application
  - To manage the privileges for a user group.

Naming Roles:-

- Within a database, each role name must be unique, and no username and role name can be the same. Unlike schema objects, roles are not "contained" in any schema.
- Therefore, a user who creates a role can be dropped with no effect on the role.

Granting and Revoking Roles:-

- You grant or revoke roles from users or other roles using the following options:
  - The Add Role to User dialog box and Remove Privileges from Role dialog box of Server Manager
  - The SQL commands GRANT and REVOKE Privileges are granted to and revoked from roles using the same options.
  - Roles can also be granted to and revoked from users using the operating system that executes Oracle.

CREATE ROLE:-

Purpose

- Use the CREATE ROLE statement to create a role, which is a set of privileges that can be granted to users or to other roles.
- You can use roles to administer database privileges. You can add privileges to a role and then grant the role to a user.
- The user can then enable the role and exercise the privileges granted by the role.

Examples

- Creating a Role: Example The following statement creates the role dw_manager:

```
CREATE ROLE dw_manager;
```

- Users who are subsequently granted the dw_manager role will inherit all of the privileges that have been granted to this role.

- You can add a layer of security to roles by specifying a password, as in the following example:

```
CREATE ROLE dw_manager
  IDENTIFIED BY warehouse;
```

- Users who are subsequently granted the dw_manager role must specify the password warehouse to enable the role with the SET ROLE statement.
- The following statement creates global role warehouse_user:

```
CREATE ROLE warehouse_user IDENTIFIED GLOBALLY
```

DROP ROLE:-

- One can easily drop the user by following statement :

```
DROP ROLE dw_manager;
```

Q-2 Write note On Creating User.

Detail :-

Introduction to Oracle CREATE USER statement:-

- The CREATE USER statement allows you to create a new database user which you can use to log in to the Oracle database.
- The basic syntax of the CREATE USER statement is as follows:

Syntax :

```
CREATE USER username
    IDENTIFIED BY password
    [DEFAULT TABLESPACE tablespace]
    [QUOTA {size | UNLIMITED} ON tablespace]
    [PROFILE profile]
    [PASSWORD EXPIRE]
    [ACCOUNT {LOCK | UNLOCK}];
```

In this syntax:

CREATE USER username

- Specify the name of the user to be created.

IDENTIFIED BY password

- Specify a password for the local user to use to log on to the database.
- Note that you can create an external or global user.

DEFAULT TABLESPACE

- Specify the tablespace of the objects such as tables and views that the user will create.
- If you skip this clause, the user's objects will be stored in the database default tablespace.

PASSWORD EXPIRE

- Use the PASSWORD EXPIRE if you want to force the user to change the password for the first time the user logs in to the database.

ACCOUNT {LOCK | UNLOCK}

- Use ACCOUNT LOCK if you want to lock user and disable access. On the other hand, specify ACCOUNT UNLOCK to unlock user and enable access.

Example :

- This example uses the CREATE USER statement to create a new local user named john with the password abcd1234:

CREATE USER john IDENTIFIED BY abcd1234;

- Oracle issues the following output indicating that user john has been created successfully.
- User JOHN created.
- Let's use the john account to log in the database.
- Launch the SQL*Plus program and enter the following information:

  Enter user-name: john@pdborcl
  Enter password:<john_password>

- Oracle issued the following error:

ERROR: ORA-01045:
user JOHN lacks CREATE SESSION privilege; logon denied

- To enable the user john to log in, you need to grant the CREATE SESSION system privilege to the user john by using the following statement:

  GRANT CREATE SESSION TO john;

- Code language:SQL (Structured Query Language)(sql)
- Now, the user john should be able to log in the database.

  Enter user-name: john@pdborcl
  Enter password:

- Connected to:
  - Oracle Database 10g Enterprise EditionRelease12.2.0.1.0 - 64b

- You can drop the user by using drop user command like:

  Drop user john;

Q-3 What is privileges?Explain Grant & Revoke statement with example.

Detail :-

- Privileges allow a user to access objects or execute programs that are owned by another user.
- These privileges can be granted (assigned) to a user or to a role.
- Granting a privilege to public is same as granting that privilege to everyone without having to specify any user name.
- In Oracle ,**Grant and Revoke statements** are used for privilege management.
- Once granted , Privileges can be revoked (cancelled) in the same manner in which they were granted.
- Oracle has three types of privileges :
  - Object Privileges
  - System Privileges
  - Role

Grant and Revoke statements with Object Privileges :-

- Object privileges are permissions on schema objects , such as tables , views , user – defined functions etc.
- With Grant option gives the grantee permission to grant the privileges to any user or role.
- These privileges enables user to perform particular action on specific object.
- Various object privileges are : ALTER DELETE , EXECUTE ,INSERT , SELECT ,UPDATE etc.

Syntax :-

GRANT <Obj privilege> ON <Object name> TO <User / Role>

Example :-

GRANT select on scott.emp to snehal;

- Object privileges can be taken back using Revoke as follow :

Syntax :-

REVOKE <Obj privilege> ON <Object name> FROM <User / Role>

Example :-

REVOKE select on scott.emp FROM snehal;

Q-4 What is Transaction ? How to Start and End the Transaction?

Detail :-

- Transaction represent an atomic,logical unit of the work.
- A Transaction in which series of SQL statements are logically related.
- In other words , a series of operations performed on Table data is known as Transaction.
- It always represent collective unit of the work.
- All the changes to the data in a Transaction are applied together or undone together.
- There are number of statements in SQL and PL/SQL that let the programmer control transactions. The programmer can :
    - Begin / Start the Transaction

- Save the Transaction
- End the Transaction

How to Start or Begin the Transaction?

- Every Transactions have a Beginning and an End.
- A Transaction begins when the first executable SQL statement is encountered.
- An executable SQL statement is a SQL statement that generates calls to Database Instance.
- A Transaction begins with an Insert , Update ,Delete or Select Statement.
- The following example execute an Update statement to begin a Transaction :

Example :-

```
Update emp set name = 'snehal' where ID = 3;
```

How to End the Transaction?

- A Transaction can end with different circumstances.
- A Transaction ends when any of the following actions occurs:
  - When user issue a Commit or Rollback Statement.
  - A user runs a DDL command such as Create , Drop , Rename or Alter.
  - A User exits normally from most oracle database utilities.
  - When any client process terminates.
  - After One Transaction Ends.

Example :-

```
Select * from emp;
Commit;
```

Q-5 Explain Commit , Savepoint and Rollback.

Detail :-

- The TCL commands are used to manage transactions in the database.
- Following are the Transaction control statements :

- Commit
- Savepoint
- Rollback

Commit :-

- The Commit Command is used to permanently save any transaction into database.
- When we use any DML command like insert , update or delete , the changes made by these commands are not permanent.
- To avoid that , we can use the commit command to mark the changes as permanent.

Syntax :-

Commit;

Example :-

Insert into emp values(1,'abc','rajkot,30);
Commit;

Savepoint:-

- The savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever required.
- In short , using this command we can name the different states of our data in any table and then rollback to that state using rollback command.

Syntax :-

Savepoint <savepoint name>

Example :-

Insert into product values(101 , 'abc' ,'rajkot');
Commit;
Update product set name = 'xyz' where name = 'abc';
Savepoint A;
Insert into product values(102 , 'pqr','surat');
Savepoint B;

Rollback :-

- This command is used to restore the database to last committed state.
- If we have used the update command to make some changes into database , and realize that those changes were not required then we can use rollback command to rollback those changes.
- **Note :-** We can use Rollback command only if they were not committed using the commit command.

Syntax :-

Rollback to <Savepoint name>

Example :-

Rollback to A;

Q-6 Give difference between SQL v/s PL/SQL

Sr.no.	SQL	PL/SQL
1	SQL is a Structural Query Language	PL/SQL is a Programming Language using SQL.
2	SQL has no variables , datatypes.	PL/SQL has variables , datatypes.
3	SQL can execute a single operation at a time.	PL/SQL can perform multiple operations at a time.
4	SQL is a declarative language.	PL/SQL is a Procedural Language
5	SQL directly interacts with database server.	PL/SQL does not directly interacts with database server.
6	SQL is a Data Oriented Language.	PL / SQL is a Application Oriented Language.
7	SQL is used to write queries and execute DML , DDL statements.	PL/SQL is used to write program blocks , functions procedures ,packages ,triggers.

Q-7 What is PL/SQL ? Explain Block Structure of PL/SQL.

Detail :-

- PL/SQL stands for Procedural Language / Structural Query Language.
- PL/SQL is the Oracle's Procedural extension to SQL with design features of Programming Language.
- It is a Block Structured Language.
- PL/SQL may have variables , constants control structures , loop structures ,error handling mechanism and so on.
- PL/SQL block structure consist mainly three section :

PL/SQL Block Structure :-

- PL/SQL is a block structure language.
- PL/SQL program may contains one or more blocks.
- Each block can be divided into three sections:
    - Declaration Section
    - Execution Section
    - Exception Section

Declaration Section :-

- Declaration Section contains Memory Variables , Constants and Other object declaration.
- This section begins with the keyword DECLARE.
- This section is optional if there is no variable / constant inside the block.

Executable Section :-

- Executable section is compulsory in PL/SQL block.
- This section begins with keyword BEGIN.
- Executable section contains SQL / PL – SQL executable statements.
- This section can have other PL/SQL blocks inside it.

Exception Section :-

- Exception section is an optional section in PL/SQL block.
- Exception section contains code to handle errors that may arise in

execution section.

- **Syntax :-**

```
DECLARE
      <Variable / Constant declaration>
BEGIN
     <Statements to be execute>
EXCEPTION
    <Statements>
END;
```

- **Example :-**

```
DECLARE
      Msg  varchar(30) := 'software development';
BEGIN
     dbms_output.put_line(msg);
END;
```

Q-8 Explain PL/SQL variable.

Detail :-

- Variable is a storage location in the memory to store the value.
- Variable must be declare before its reference in execution section or in exception section.

- **Syntax :-**

```
<Variable_name> <data type> := <value>;
```

- Any variable /constant declared NOT NULL , must be given value immediately using assignment operator (:=).
- Multiple variables cannot be declared in the same statement.
- With Variable name , the constant must be initialized.

- Variable must be declare inside declaration section of PL/SQL block;
- Variable can be of any type like , number ,character , date , Boolean etc.

- **Example :-**

  Name varchar2(30) := 'snehal pandya';

Q-9 Explain PL/SQL Data type.

Detail :-

- Data type is used to decide the type of data.
- The following is a list of data type available in PL/SQL.
  - Scalar Data type
  - LOB – Large Object Data type
  - Composite Data type
  - Reference Data type

1. Scalar Data type :-

- The Scalar data type include mainly the basic or common data types.
- Scalar data type include mainly the following :
  - Numeric
  - Character
  - Boolean
  - Date

Numeric :-

- **Numeric Data type include mainly the following:**
  - **Number(Prec , Scale)** :- It include the numbers with or without decimal points.
  - **Real :-** It include Floating point type with maximum precision of 63 digits.
  - **Integer :-** It include integer type with maximum precision of 38 digits.
  - **Decimal(Prec , Scale) :-** It includes fixed point type with maximum precision of 38 decimal digits.

Character:-

- **Character Data type include mainly the following:**
  - **Char :-** It is fixed length character string with maximum size of 32,767 bytes.
  - **Varchar2 :-** It is variable length character string with maximum size of 32,767 bytes.
  - **Long :-** It is variable length character string with maximum size of 32,760.
  - **RowID :-** It is physical row identifier , the address of row in an ordinary table.

Boolean :-

- The boolean data type stores logical values that are used in logical operations.
- The logical values are the boolean values that is either **True** or **False**.

Date :-

- The date data type is used to store fixed length date times with valid range from January 1,4712 BC to December 31,9999 AD.
- The Date format should be DD – MON – YY.

2. LOB – Large Object Data type :-

- The Lob data typte is mainly used to store and manipulate large block of unstructured Data like : Images , Multimedia Files etc.
- Lob data types includes mainly the following :
  - BLOB
  - CLOB
  - BFILE

BLOB :-

- This data type stores the LOB data in the binary file format upto maximum size of 128 TB.

CLOB :-

- This data type stores the LOB data into the character set with the

maximum size upto 128 TB.

BFILE :-

- This data type is used to store unstructured binary format data outside the database.
- The size of BFILE is to a limited operating system and they are read only files and can not be modified.

3. Composite Data type :-

- The Composite Data type is a data type that having internal components that can be manipulated individually.
- Composite data types includes mainly the following elements :
  - Array
  - Record
  - Table
  - Nested Table
  - List
  - Collection

4. Reference Data type :-

- This data type is used to hold pointer value.
- Pointer value as a reference , just stores address of other program items.

Q-10 Explain PL/SQL Control Structure.

Detail :-

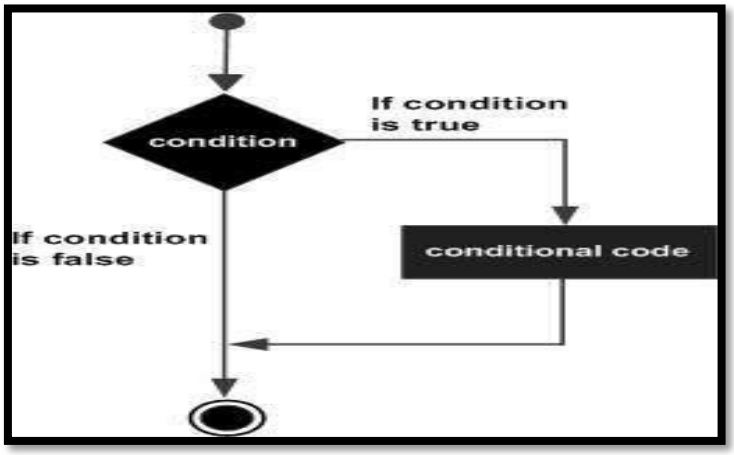
- In PL/SQL , the control structure provide facility to specify one or more conditions at the same time.
- The control statements return the result or decision on the basis of any condition that can be true or false.
- PL/SQL supports mainly the following control structure:
  - IF statement
  - CASE statement

1. IF STATEMENT :-

- The If statement provide facility to check the condition and return some result based on that condition.
- If statement is one of the powerful Control Structure that support decision capability.
- If statement having mainly the following Flavors :
  - IF…Then
  - If…Then…Else
  - If…Then…Elsif
  - Nested If…Then…Else

1. IF…Then :-

- The if statement is used to give condition and return the result.
- In this case , if the condition is true then statement is executed .
- But if condition become false then if will be terminated.



Syntax :-

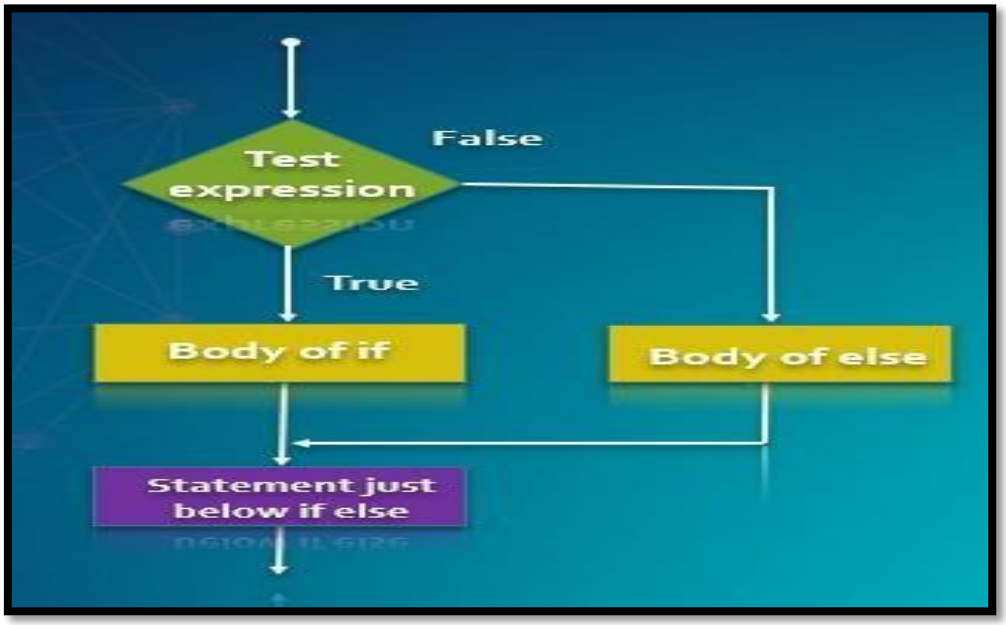
```
If condition then
        <statements>
End if;
```

Example :-

```
If (a<=20) then
        c:=c+1;
End if ;
```

2. IF…Then :-

- In this statement , First of all condition will be checked.
- If condition become true , then the statement following if will be execute.
- But if condition become false , then the statement following else will be execute.



Syntax :-

```
If condition then
        <statements>
Else
        <statements>
End if;
```

Example :-

```
If (a<=20) then
        c:=c+1;
else
        b:=b+1;
End if;
```

3. IF…Then…Elsif :-

- The If..Then..Elsif statement allows you to give multiple conditions at the same time.
- First condition will be tested first , If first condition become false then other conditions will be tested.
- At a time only one condition become true.
- If no any condition become true then finally the statement following else Will be execute.

Syntax :-

```
If condition then
        <statements>
Elsif condition then
        <statements>
Elsif condition then
        <statements>
End if;
```

Example :-

```
If (a<=20) then
        c:=c+1;
elsif(a>=20) then
        b:=b+1;
else
        d:=d+1;
End if;
```

4. Nested If :-

- When you create one if statement inside another if statement then it is called nested if structure.
- First of all outer if condition will be checked , it it become true then inner if condition will be checked.
- If outer if condition become false then finally the statement following else of outer if will be execute.

Syntax :-

```
If condition then
        <statements>

        If condition then
            <statements>
        Else
            <statements>
        End if;
Else
        <statements>
End if;
```

2. CASE STATEMENT :-

- Like if statement the CASE statement provide facility to apply multiple conditions at the same time.
- It is mainy used for multiple selection at the same time.
- The CASE statement select one sequence of statements to execute.
- To select the sequence , A CASE statement use a selector rather than multiple Boolean expressions.
- A selector is an expression , the value of which is used to select one of several alternatives.

Syntax :-

```
CASE selector
    WHEN 'value1' THEN
        <statement-1>;
    WHEN 'value1' THEN
        <statement-2>;
    WHEN 'value1' THEN
        <statement-3>;

    Else [Default Case]
        <statement>
END CASE;
```

Example :-

```
DECLARE
      Ch char(1) := 'a';
BEGIN
      Case ch
            When  'a' then
                Dbms_output.put_line('it is a');
            When  'b' then
                Dbms_output.put_line('it is b');
            When  'c' then
                Dbms_output.put_line('it is c');
            Else
                Dbms_output.put_line('invalid');
       END CASE;
  END;
```

Q-11 Explain PL/SQL Looping / Iterative Structure.

Detail :-

- In PL/SQL ,Looping statements are used to execute or repeat the statements for number of times.
- In general , the statements are executed sequentially.
- The loop statements allowed to execute group of statements multiple time until the given condition become false.
- In PL/SQL , following types of loops available:
    - Basic Loop
    - While Loop
    - For Loop

1. Basic Loop :-

- The basic loop structure provide facility to execute group of statements.
- It enclose statements between Loop and End Loop.
- With each iteration , the sequence of statements is executed and then

control resumes at the top of the loop.

Syntax :-

```
Loop
        <statements>
End loop;
```

Example :-

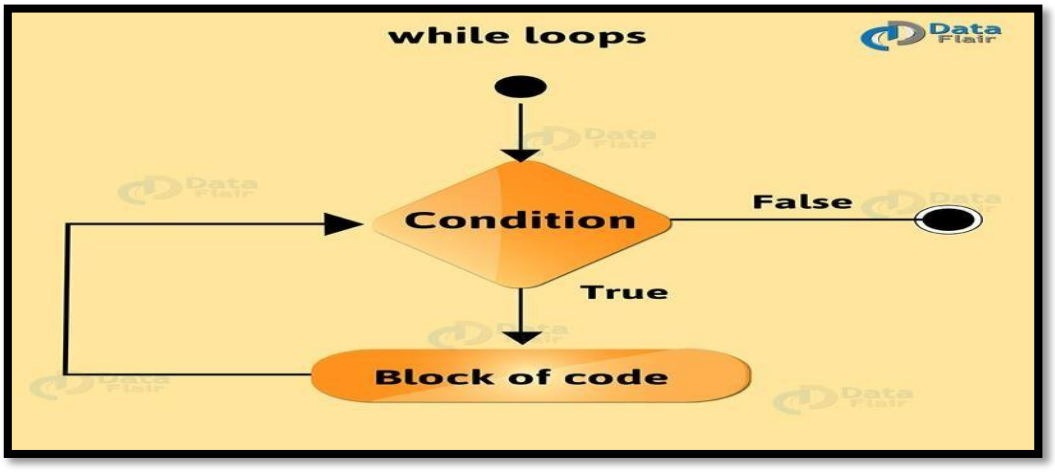
```
DECLARE
    X number :=10;

BEGIN
    Loop
            dbms_output.put_line(x);
            x:=x+10;
    End loop;
END;
```

Note :- An EXIT statement or an EXIT WHEN statement is required to break the loop.

2. While Loop :-

- A While loop statement in PL/SQL provide facility to execute group of statements repeatedly as long as a given condition is true.
- When condition become false , the loop will be terminated.



Syntax :-

```
While <condition> Loop
        <statements>
End loop;
```

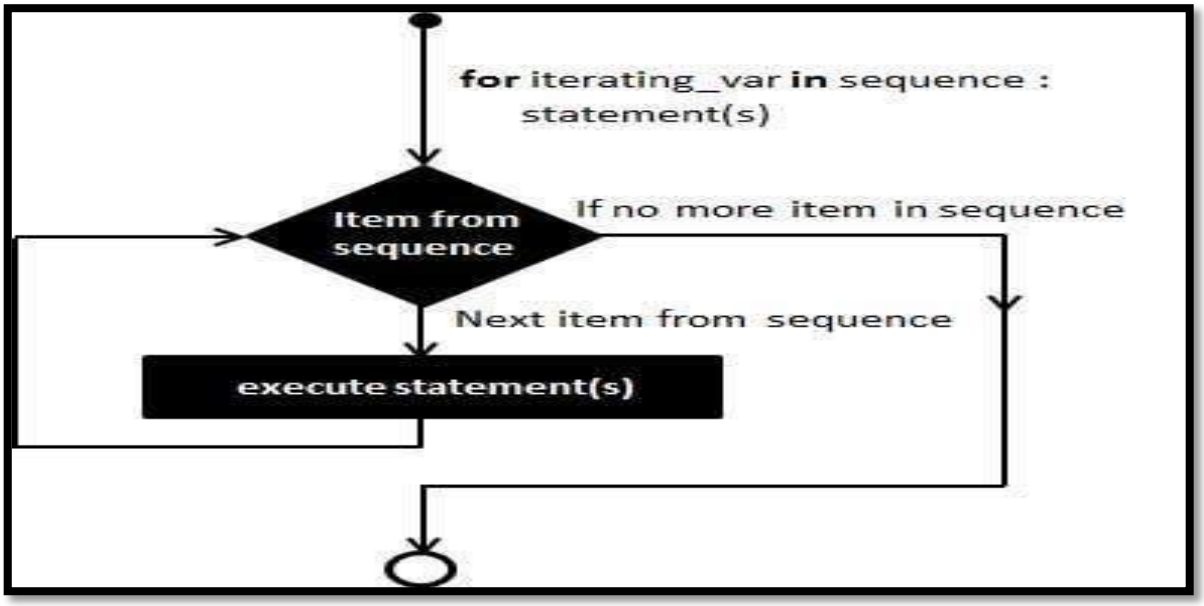
Example :-

```
DECLARE
    X number (2)  :=10;

BEGIN
    While x < 20 Loop
            dbms_output.put_line(x);
            x:=x+1;
    End loop;
END;
```

3. For Loop :-

- A For loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- First of all , loop variable will be initialized , then condition will be checked and then finally statements will be execute.



Syntax :-

```
For variable in  initial_value .. final_value Loop
        <statements>
End Loop;
```

Example :-

```
DECLARE
    X number (2)  :=10;
BEGIN
      FOR  X in 10 .. 20 LOOP
            Dbms_output.put_line(a);
       END LOOP;
 END;
```

Q-12 Explain Cursor with Types and Example.

Detail :-

- Oracle Creates a memory area , that known as context area.
- A cursor is a pointer to this context area.
- PL/SQL controls the context area through a cursor.
- A cursor holds the rows(one or more) returned by SQL statement.
- The set of the rows the cursor holds is known as the active set.
- You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statements.
- There are two types of cursors available:
    - Implicit Cursor
    - Explicit Cursor

1. Implicit Loop :-

- The implicit cursors are automatically generated by oracle while an SQL statement is executed.
- Implicit cursors are created by default to process the statements when

DML statements like INSERT , UPDATE ,DELETE etc. are executed.

- Oracle provide some attributes of implicit cursors which are as follow:
    - **%FOUND :-** It returns TRUE if DML statements like INSERT UPDATE ,and DELETE affect at least one or more rows.
    - **% NOTFOUND :-** It returns TRUE if DML statements like INSERT UPDATE ,and DELETE affect no row.
    - **%ISOPEN :-** It always returns false for implicit cursors , because SQL cursor is automatically closed after executing Statements.
    - **%ROWCOUNT :-** It returns the number of rows affected by DML statements like INSERT UPDATE ,and DELETE.

2. Implicit Loop :-

- The Explicit cursors are defined by the programmers to gain more control over the context area.
- These cursors should be defined in the declaration section of the PL/SQL block.
- You must follow these steps while working with an explicit cursor.
    - Declare the cursor to initialize the memory.
    - Open the cursor to allocate the memory.
    - Fetch the cursor to retrieve data.
    - Close the cursor to release allocated memory.

1. **Declare the Cursor :-** It defines the cursor with a name and the associated select statement.
   Ex :- CURSOR name IS SELECT statement

2. **Open the Cursor :-** It is used to allocate memory for the cursor and make it easy to fetch the rows returned by the SQL statements into it.
   Ex :- OPEN cursor_name

3. **Fetch the Cursor :-** It is used to access one row at a time. You can fetch rows from the opened cursor.
   Ex:- FETCH cursor_name INTO variable_list;

4. **Close the Cursor :-** It is used to release the allocated memory and close

the opened cursors.

Ex:- Close cursor_name

Example :- (Explicit Cursor)

```
DECLARE
        Empid emp.id%type;
        Ename emp.name%type;
        Eage emp.age%type;

CURSOR  e_cursor is
    SELECT id,name,age from emp;

    BEGIN
        OPEN e_cursor;
        LOOP
            FETCH e_cursor into Empid ,Ename,Eage;
            Exit when e_cursor% NOTFOUND;

        Dbms_output.put_line(Empid || Ename || Eage);
        END LOOP;

        CLOSE e_cursor;

    END;
/
```

Q-13 Write note on %Type and %Rowtype

Detail :-

- The PL/SQL datatype support %Type and %Rowtype.

1.%Type :-

- To declare variable of the same data type dynamically , as the database column , we may use %type declration.

- This is called dynamic data type assignment.
- It is used to reference the data type and size of the table or view in the data base.
- The datatype and size will be inherited from column.
- This is important , as we need not change the program code if the data type or length of the table column is changed.
- The changes to the size or type in the table are automatically reflected in the program.
- In %Type , you can specify column names , variable , record , cursor or constant declaration.

Example :-

Empno employee.emp_id % Type;
Empnm employee.emp_name%Type;

2.%RowType :-

- PL/SQL gives you explicit and implicit techniques for defining records.
- Using the %RowType attribute , a record variable can be declared implicitly based on the structure of a table or query.
- In implicitly defined record , we don't have to describe each field in the record definition.
- The changes to the underlying table structure result in automatic changes to the record.

Example :-

Acc_info account %Rowtype;